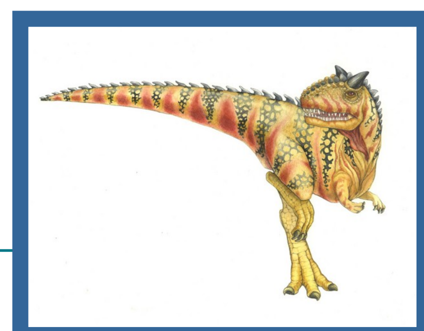
Operating System Concepts

(slides adapted from 10th ed. Silberschatz, Galvin and Gagne)

Chapter 16

Security



Objectives

- Discuss security **threats and attacks**
- Explain the fundamentals of encryption, authentication, and hashing
- Examine the uses of **cryptography** in computing
- Describe the various **countermeasures** to security attacks

JE HOEFT GEEN "FORMULES" TE KENNEN
→ zorg dat je concepten in woorden kan uitleggen!

The Security Problem

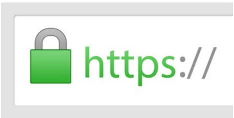
- System **secure** if resources used and accessed as intended under all circumstances
  - Unachievable
- **Intruders** (**crackers**) attempt to breach security
- **Threat** is potential security violation
- **Attack** is attempt to breach security
- Attack can be accidental or malicious
- Easier to protect against accidental than malicious misuse

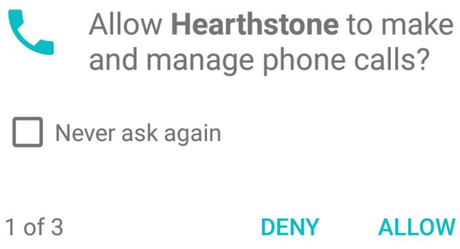
"safety" vs. "security": presence of an intelligent adversary!

Security Violation Categories: “CIA Triad”

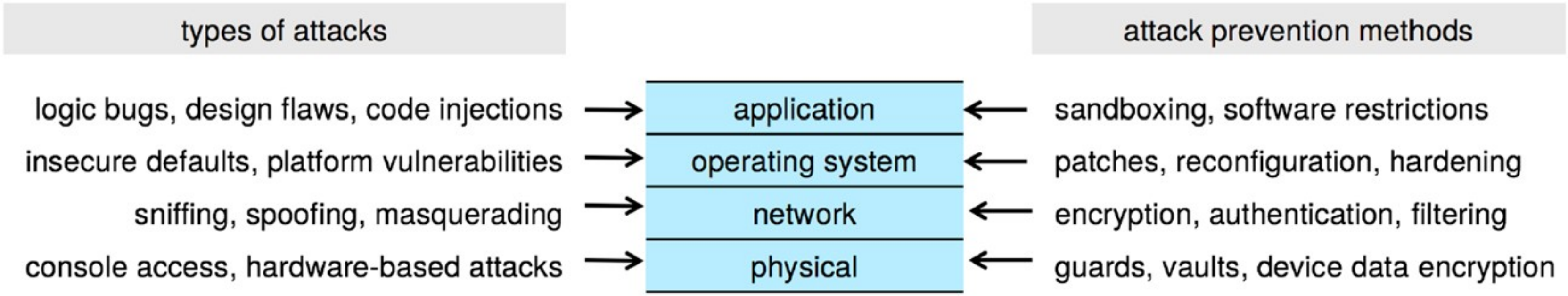
Security Measure Levels

- Impossible to have absolute security, but “raise the bar” to deter most intruders
- Security must occur at four levels to be effective:
  - **Physical**
    - Data centers, servers, connected terminals
  - **Network**
    - Intercepted communications, interruption, DOS
  - **Operating System**
    - Protection mechanisms, updates
  - **Application**
    - Benign or malicious apps may contain security bugs



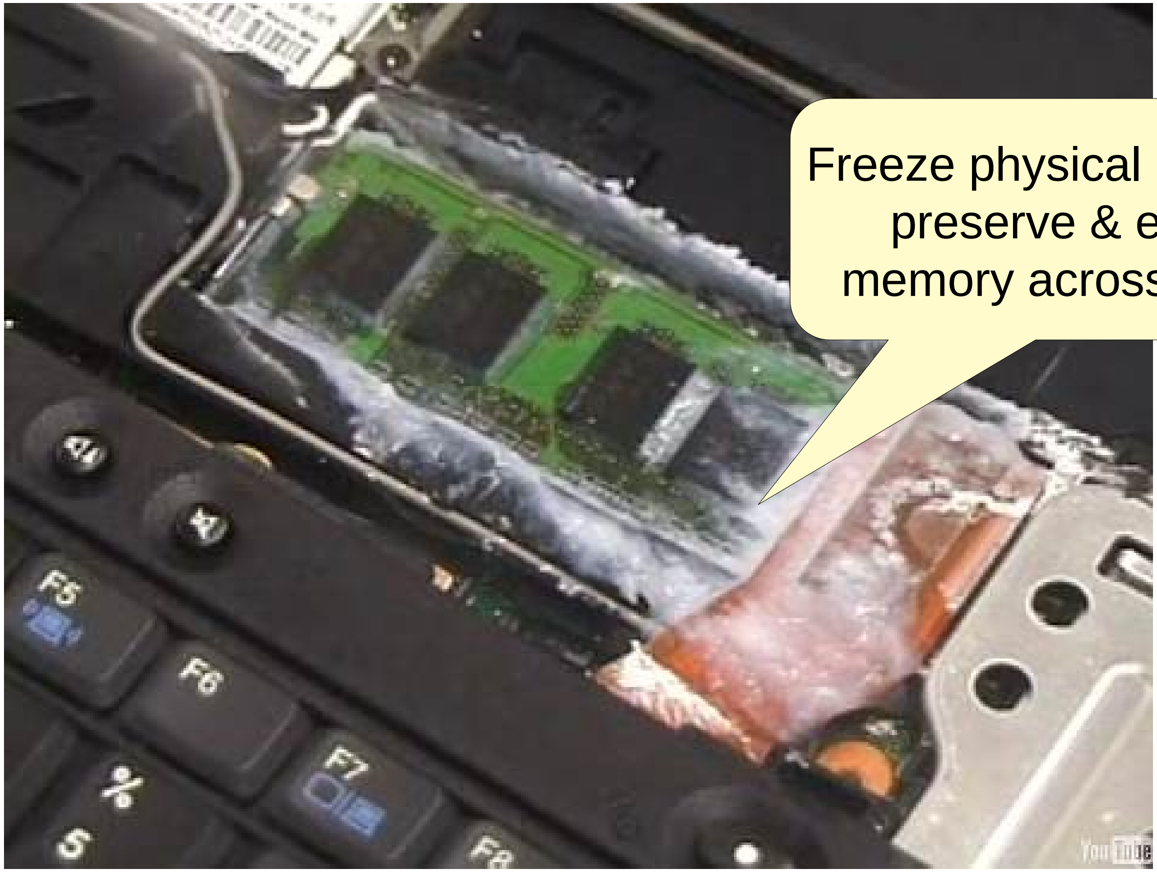


Four-layered Model of Security



Security is as weak as the weakest link in the chain!

Example: Cold-Boot Attacks



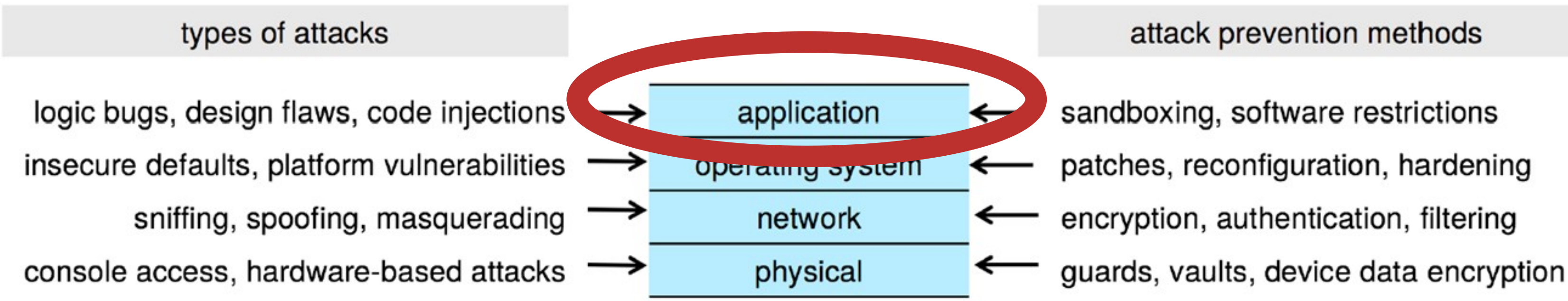
Freeze physical DRAM to preserve & extract memory across reboots

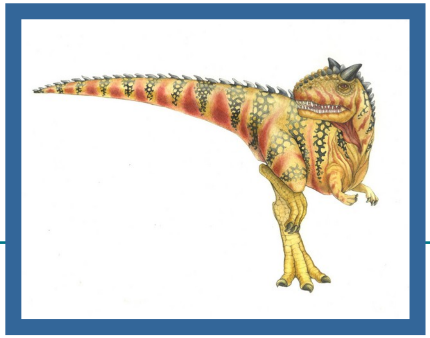
Security is as weak as the weakest link in the chain!

https://www.ethicalhacker.net/features/root/using-cold-boot-attacks-forensic-techniques-penetration-tests/

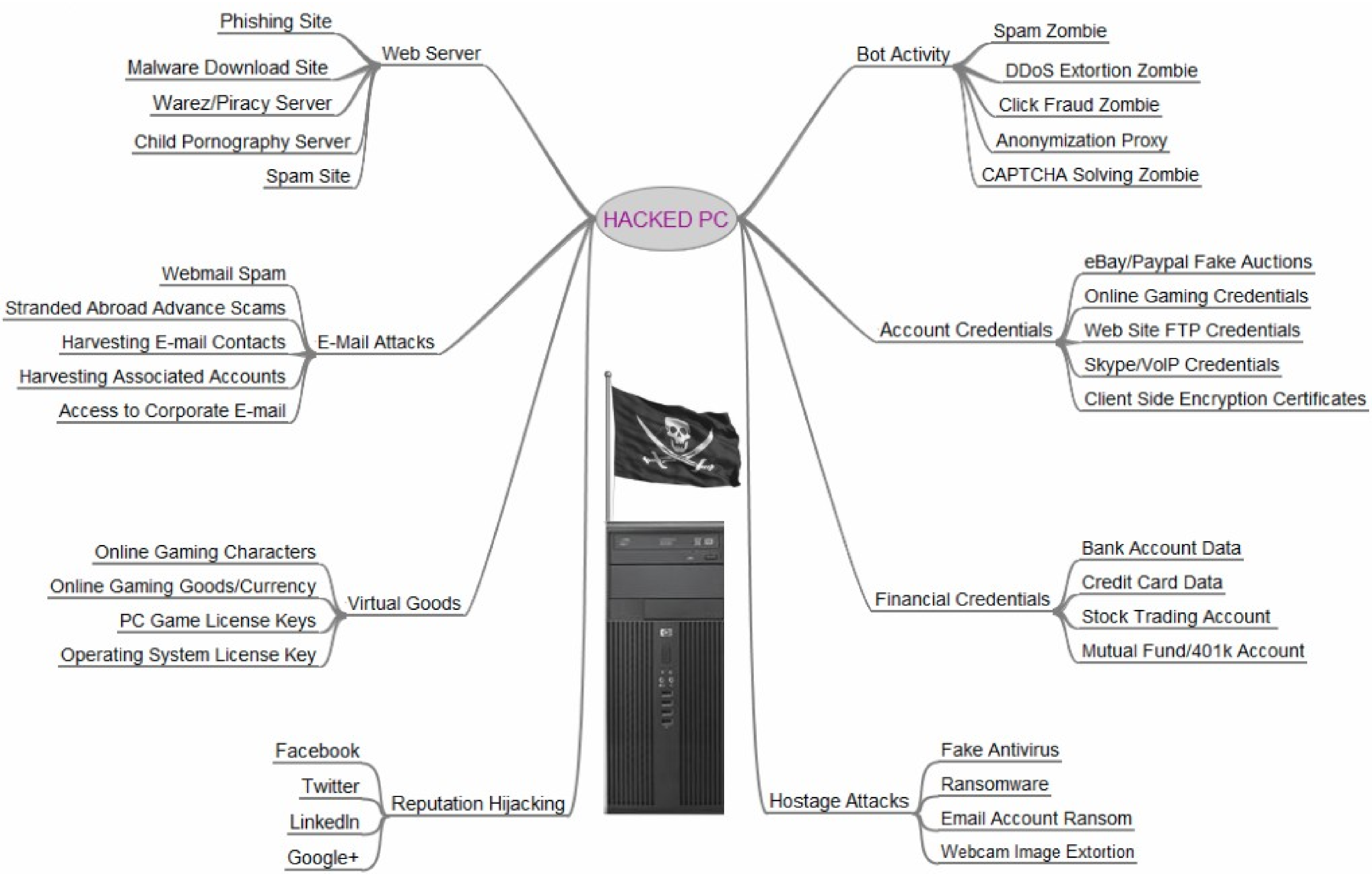
https://www.thewindowsclub.com/cold-boot-attack

16.2 Program Threats





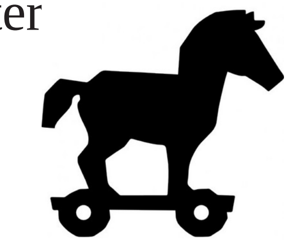
Program Threats: Motivation



https://krebsonsecurity.com/2012/10/the-scrap-value-of-a-hacked-pc-revisited/

Program Threats: Classification

- **Malware -** Software designed to exploit, disable, or damage computer
- **Trojan Horse –** Program that acts in a clandestine manner
  - **Spyware** – Program frequently installed with legitimate software to display adds, capture user data
  - **Ransomware** – locks up data via encryption, demanding payment to unlock it
- **Trap/Back Door, Logic Bomb**
  - e.g., specific user ID or password circumvents normal security procedures
  - Difficult to detect: **code review**
  - … but could even be included in a compiler to propagate invisibly (!)



cf. Ken Thompson’s Turing Award Lecture: “Reflections on Trusting Trust”

Principle of Least Privilege – (or why do Trojans thrive?)

"Every program and every privileged user of the system should operate using the least amount of privilege necessary to complete the job."

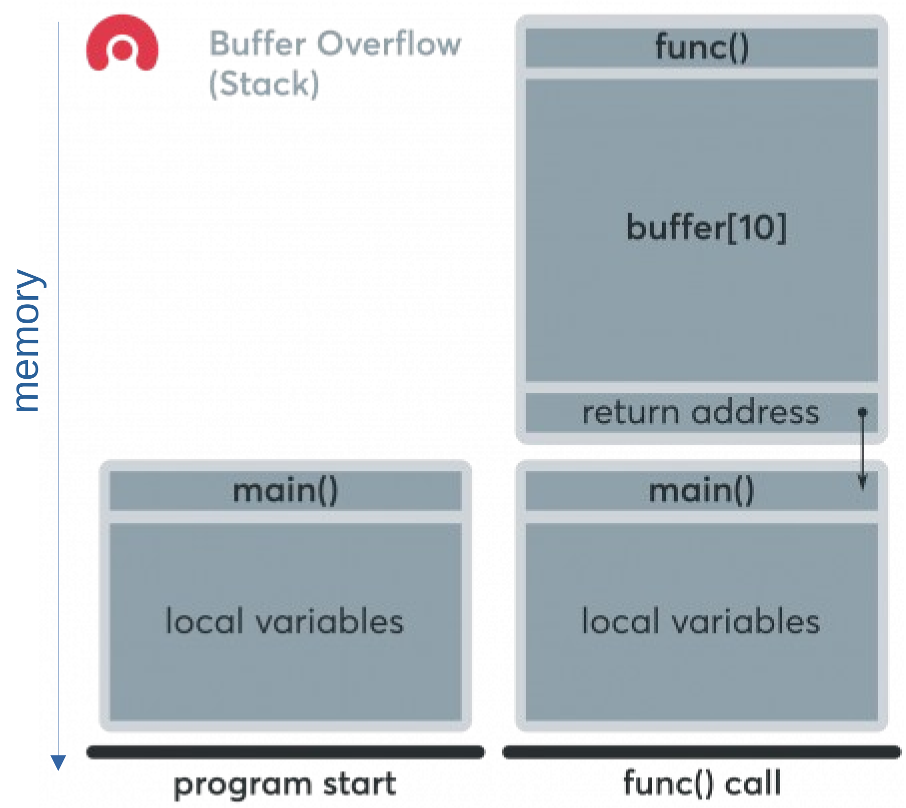
— Jerome H. Saltzer, 1974

Program Threats: Code Injection

- **Code-injection attack** occurs when system code is not malicious but has bugs allowing *executable code to be added or modified*
  - Results from poor or insecure programming paradigms, commonly in low level languages like C or C++ which allow for direct memory access through pointers
  - Goal is a **buffer overflow** in which code is placed in a buffer and execution caused by the attack

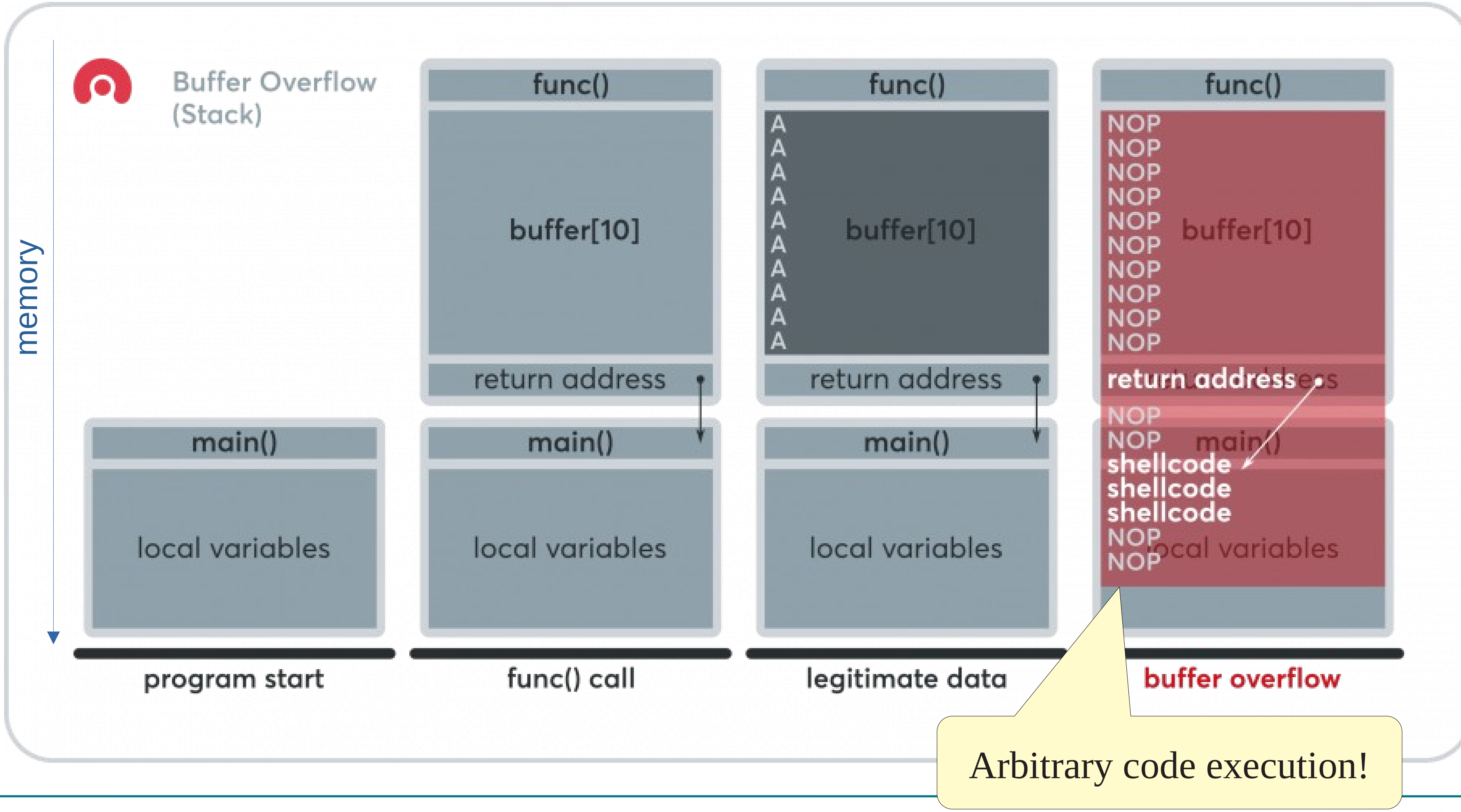
C Program with Buffer-overflow Condition

```
void func(char *s) {
    char buffer[10];
    strcpy(buffer, s);
}

int main(int argc, char *argv[]) {
    if (argc < 2)
        return -1;

    func(argv[1]);
    return 0;
}
```



- **Code review** can help – programmers review each other’s code, looking for logic flows, programming flaws

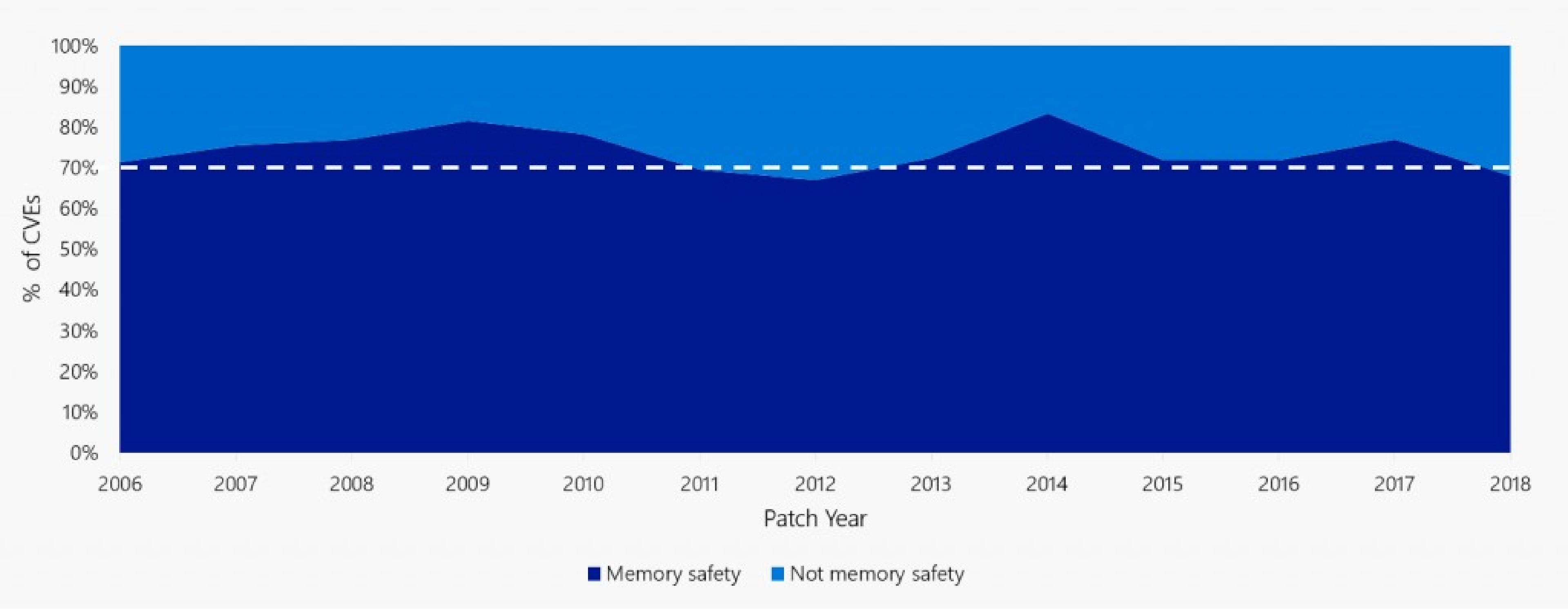
C Program with Buffer-overflow Condition



Great Programming Required?

- For the first step of determining the bug, and second step of writing exploit code, yes
- **Script kiddies** can run pre-written exploit code to attack a given system
- Attack code can get a shell with the processes' owner's permissions
  - Or open a network port, delete files, download a program, etc
- Depending on bug, attack can be executed across a network using allowed connections, bypassing firewalls
- Various mitigations at compiler, operating system levels
  - E.g., execute-disable bit in Page Table Entries (cf. xv6-riscv exercise session)
  - Raising the bar, nothing perfect for C/C++ → use **safe languages** (eg Rust, Java, Python, etc.)!

Memory-safety issues remain dominant!



https://github.com/microsoft/MSRC-Security-Research/blob/master/presentations/2019_02_BlueHatIL/2019_01%20-%20BlueHatIL%20-%20Trends%2C%20challenge%2C%20and%20shifts%20in%20software%20vulnerability%20mitigation.pdf

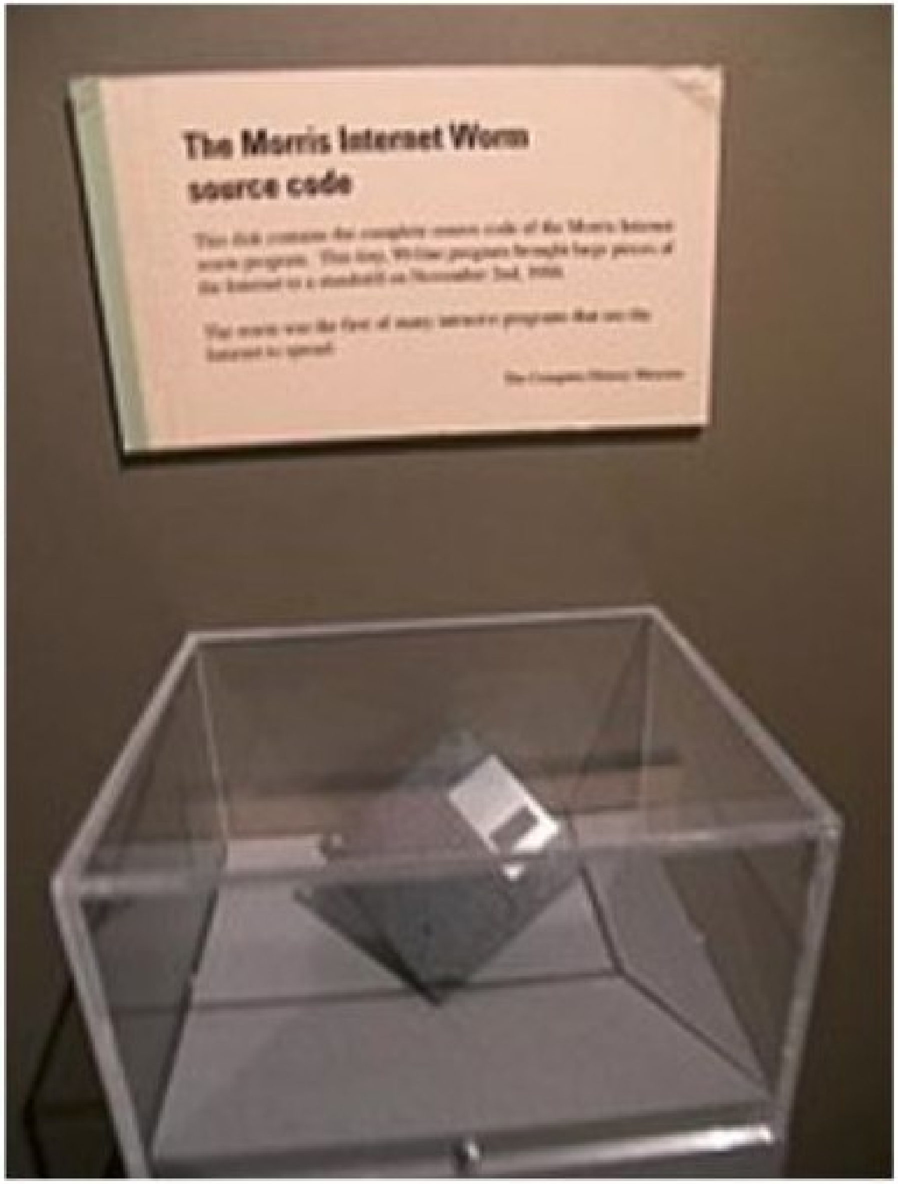
Program Threats 3

Attacker-defender race: Adapt and stay under the radar of **anti-virus software**

- **Viruses**
  - *Self-replicating*, designed to infect other computers
  - Very specific to CPU architecture, operating system, applications
  - Usually borne via email or as a macro
  - Visual Basic Macro to reformat hard drive

```
Sub AutoOpen()
Dim oFS
  Set oFS = CreateObject(''Scripting.FileSystemObject'')
  vs = Shell(''c:command.com /k format c:'',vbHide)
End Sub
```

Morris Worm: The first Internet virus infection

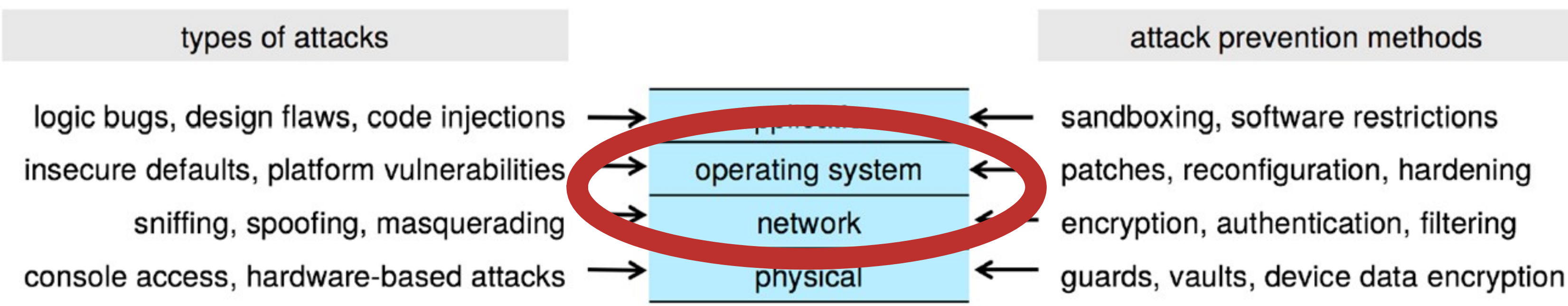


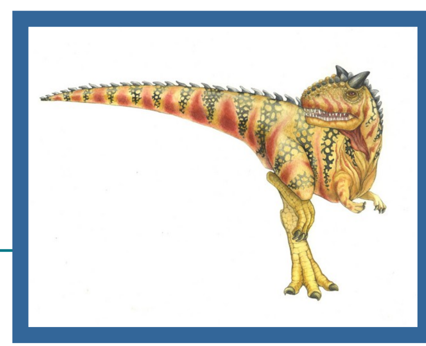
https://en.wikipedia.org/wiki/Morris_worm

The Threat Continues

- Attacks still common, still occurring
- Attacks moved over time from science experiments to tools of organized crime
  - Targeting specific companies
  - Creating botnets to use as tool for spam and DDOS delivery
  - **Keystroke logger** to grab passwords, credit card numbers
- Why is Windows the target for most attacks?
  - Most common
  - Everyone is an administrator
    - Licensing required?
  - **Monoculture** considered harmful

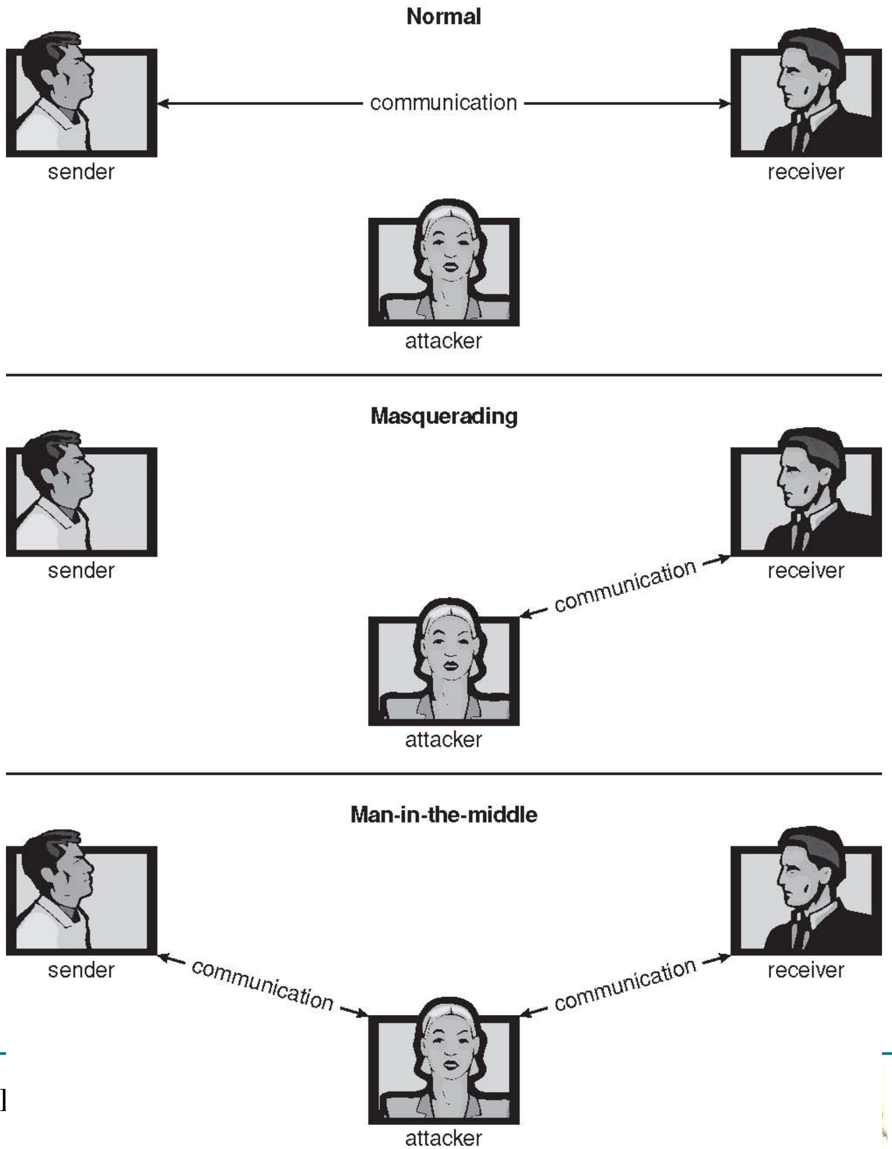
16.3 System & Network Threats





Attacking Network Traffic

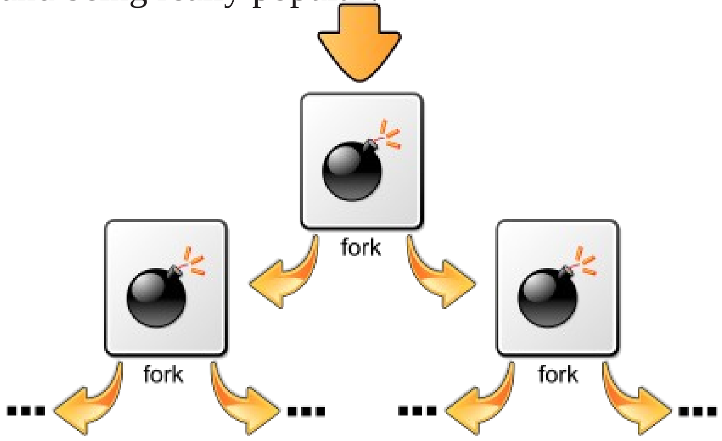
- **Masquerading** (breach authentication)
  - Pretending to be an authorized user to **escalate privileges**
- **Replay attack**
  - As is or with **message modification**
- **Man-in-the-middle attack**
  - Intruder sits in data flow, masquerading as sender to receiver and vice versa



Denial of Service

- Overload the targeted computer preventing it from doing any useful work
- **Distributed Denial-of-Service** (**DDoS**) come from multiple sites at once
- Consider the start of the IP-connection handshake (SYN)
  - How many started-connections can the OS handle?
- Consider traffic to a web site
  - How can you tell the difference between being a target and being really popular?
- Accidental – CS students writing bad `fork()` code
- e.g., “**Fork bomb**”

Bash shell one-liner: “:(){ :|:& };:”



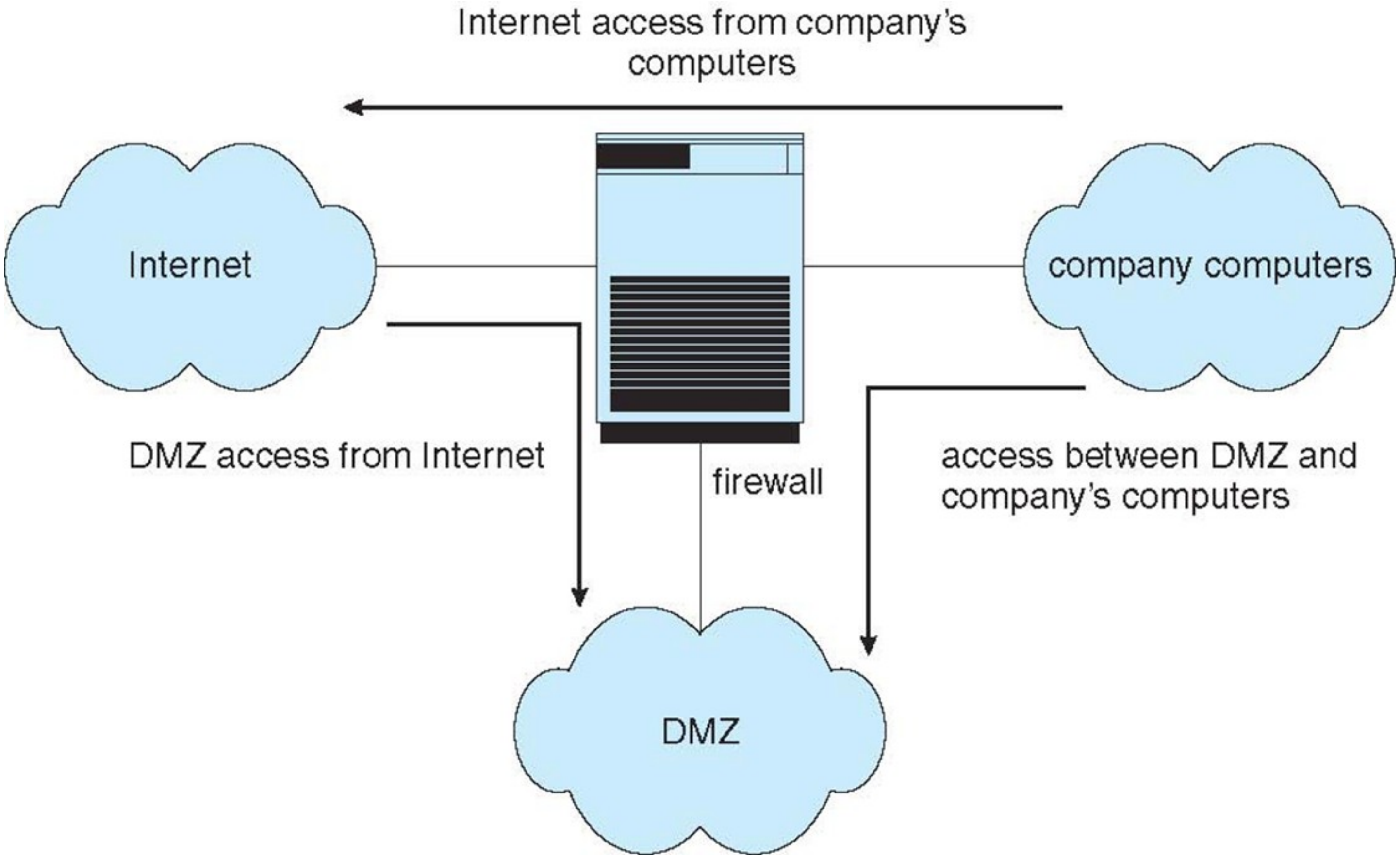
Port Scanning

- Automated attempt to connect to a range of ports on one or a range of IP addresses
- Detection of answering service protocol
- Detection of OS and version running on system
- `nmap` scans all ports in a given IP range for a response
-

jo@breuer: ~/Documents/doc/presentations

```
jo@breuer:~/Documents/doc/presentations$ sudo nmap -O scanme.nmap.org

Starting Nmap 7.60 ( https://nmap.org ) at 2021-11-23 14:25 CET
Nmap scan report for scanme.nmap.org (45.33.32.156)
Host is up (0.15s latency).
Other addresses for scanme.nmap.org (not scanned): 2600:3c01::f03c:91ff:fe18:bb2f
Not shown: 991 closed ports
PORT      STATE    SERVICE
22/tcp    open     ssh
25/tcp    filtered smtp
53/tcp    filtered domain
80/tcp    open     http
135/tcp   filtered msrpc
139/tcp   filtered netbios-ssn
445/tcp   filtered microsoft-ds
9929/tcp  open     nping-echo
31337/tcp open     Elite
Aggressive OS guesses: Linux 2.6.32 (89%), Linux 3.4 (89%), Linux 3.5 (89%), Linux 4.2 (89%
), Synology DiskStation Manager 5.1 (89%), Linux 3.10 (88%), Linux 2.6.32 or 3.10 (88%), Li
nux 4.4 (88%), WatchGuard Fireware 11.8 (88%), Linux 3.1 - 3.2 (88%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 10 hops

OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.57 seconds
jo@breuer:~/Documents/doc/presentations$
```

Network Security Through Domain Separation Via Firewall

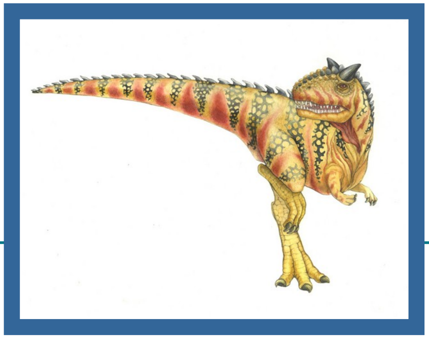


Firewalling to Protect Systems and Networks

- A network **firewall** is placed between trusted and untrusted hosts
  - The firewall limits network access between these two **security domains**
- Can be tunneled or spoofed
  - Tunneling allows disallowed protocol to travel within allowed protocol (i.e., telnet inside of HTTP)
  - Firewall rules typically based on host name or IP address which can be spoofed
- **Personal firewall** is software layer on given host
  - Can monitor / limit traffic to and from the host
- **Application proxy firewall** understands application protocol and can control them (i.e., SMTP)
- **System-call firewall** monitors all important system calls and apply rules to them (i.e., this program can execute that system call)

16.4 Cryptography as a Security Tool

JE HOEFT GEEN “FORMULES” TE KENNEN
→ zorg dat je concepten in woorden kan uitleggen!

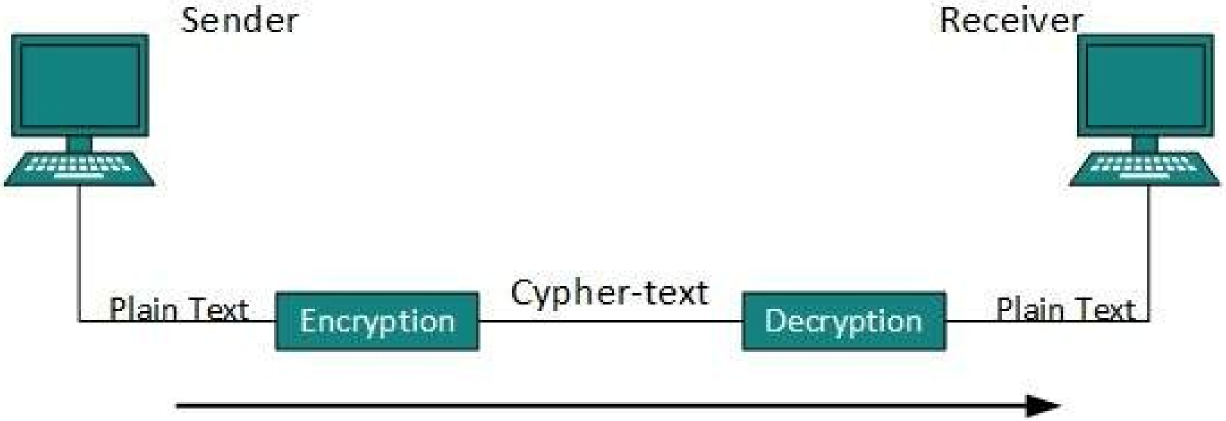


Cryptography as a Security Tool

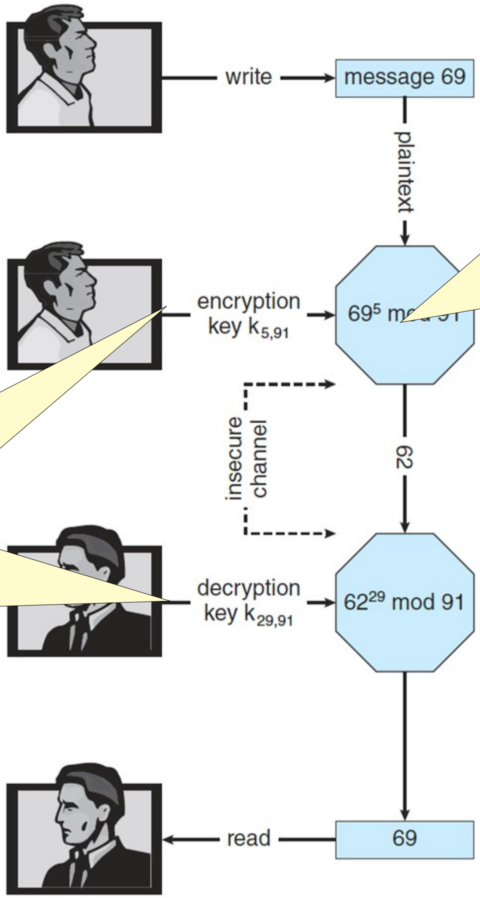
- Broadest security tool available
  - Internal to a given computer, source and destination of messages can be known and protected
    - OS creates, manages, protects process IDs, communication ports
  - Source and destination of messages on network cannot be trusted without cryptography
    - Local network – IP address?
      - Consider unauthorized host added
    - WAN / Internet – how to establish authenticity
      - Not via IP address

Cryptography 1

- Means to constrain potential senders (*sources*) and / or receivers (*destinations*) of *messages*
  - Based on secrets (**keys**)
  - Enables
    - Confirmation of source == **Authentication**
    - Receipt only by certain destination == **Encryption/decryption**
    - Trust relationship between sender and receiver



Secure Communication over Insecure Medium



No worries,
focus on **concepts**!

→ Consider encryption and decryption as "black boxes"

Sender & receiver can share the same key

= **symmetric** crypto

Or not
= **asymmetric** crypto

Encryption [1]

- Constrains the set of possible receivers of a message
- **Encryption** algorithm consists of
  - A function to encrypt $E(K, M) \rightarrow C$. That is, C is the ciphertext for a given key K and plaintext message M
  - A function to decrypt $D(K, C) \rightarrow M$. That is, M is the plaintext for a given K and ciphertext C

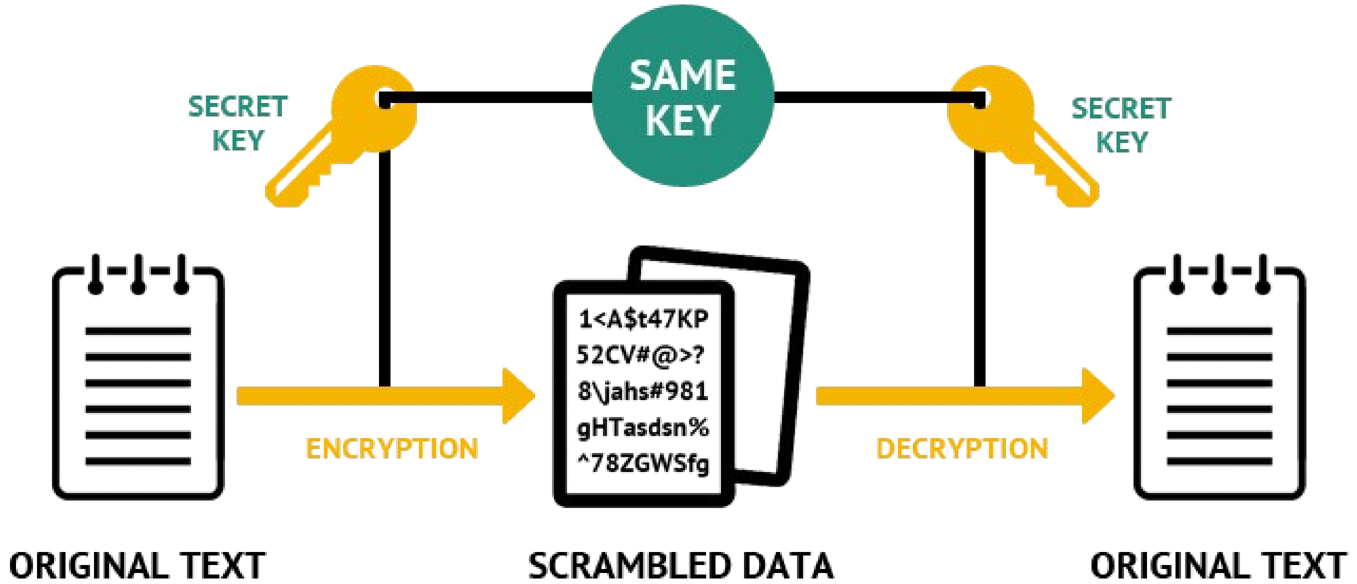
Essential property: Given a ciphertext c, a computer can compute m such that

E(K, m) = c and *only* if it possesses K.

Thus, the only way to construct m is D(K,c) = m, and a computer not holding K cannot decrypt ciphertexts

Symmetric Encryption (e.g., AES, DES)

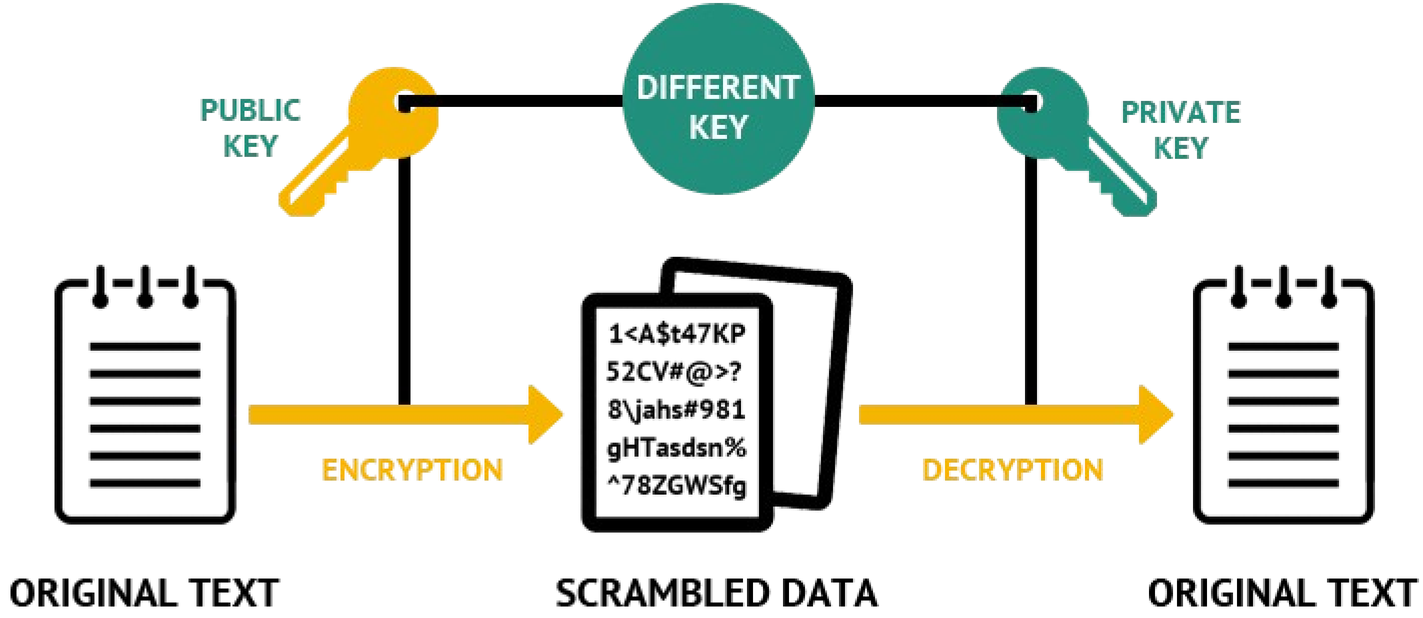
- <u>Same key </u>used to encrypt and decrypt
  - *E*(*k*) can be derived from *D*(*k*), and vice versa
  - Main problem: **key distribution**
- Most widely used: Advanced Encryption Standard (**AES**), supersedes (triple) **DES**



https://www.cheapsslshop.com/blog/symmetric-vs-asymmetric-encryption-whats-the-difference

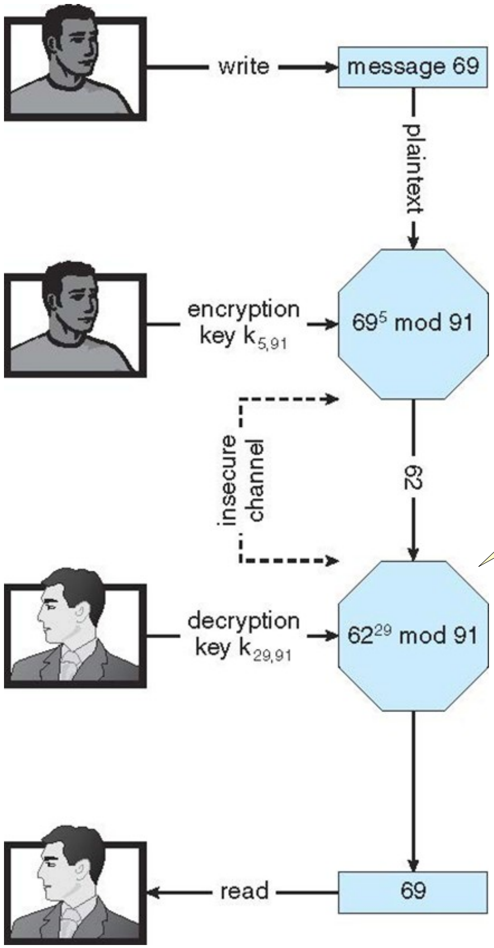
Asymmetric Encryption (e.g., RSA)

- **Public-key encryption** based on each user having two keys:
  - **public key** – published key used to encrypt data
  - **private key** – key known only to individual user used to decrypt data



https://www.cheapsslshop.com/blog/symmetric-vs-asymmetric-encryption-whats-the-difference

Encryption using RSA Asymmetric Cryptography



Focus on the **concepts**.

You don't need to know RSA details!

Cryptography 2

- Note **symmetric** cryptography based on transformations, **asymmetric** based on mathematical functions
  - Asymmetric much more compute intensive
  - Typically not used for bulk data encryption

cf. key establishment in SSL/TLS (later)

Authentication [1]

- Constraining set of potential senders of a message
  - Complementary to encryption
  - Also can prove message unmodified

Confidentiality without **integrity** (mostly) not very useful

Remember the CIA triad!

- Authentication function $S(K, M) \rightarrow A$ That is a function for generating authenticators from messages
- Verification function $V(K, M, A) \rightarrow \{\text{true}, \text{false}\}$. That is, a function for verifying authenticators on messages

Building Block – Hash Functions

- Basis of authentication
- Creates small, fixed-size block of data **message digest** (**hash value)** from *m*
- Hash Function *H* must be collision resistant on *m*
  - Must be infeasible to find an $m' \neq m$ such that $H(m) = H(m')$
- If $H(m) = H(m')$, then $m = m'$
  - The message has not been modified
- E.g., **MD5**, **SHA-3**
- Not useful as authenticators
  - For example *H(m)* can be sent with a message
    - But if *H* is known someone could modify *m* to *m'* and recompute *H(m')* and modification not detected
    - So must authenticate *H(m)*

== One-way "compression" function

Authentication - MAC

- Symmetric encryption used in **message-authentication code** (**MAC**) authentication algorithm
- Cryptographic checksum generated from message using secret key
  - Can securely authenticate short values
- If used to authenticate *H(m)* for an *H* that is collision resistant, then obtain a way to securely authenticate long message by hashing them first!

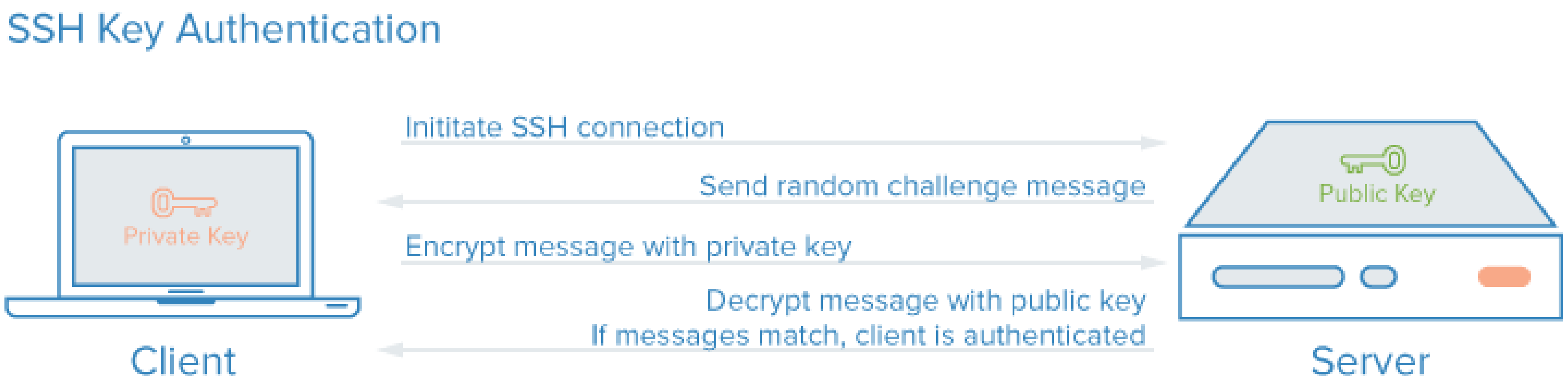
Authentication – Digital Signature [1]

- Based on asymmetric keys and digital signature algorithm
- Authenticators produced are **digital signatures**
- Very useful – ***anyone*** can verify authenticity of a message

Example (RSA): Consider public key E and secret key S and hash function H:

- Sign message M: *encrypt(S,H(M)) = A* → *sign with private key*
- Verify signature A: *decrypt(E,A) = H(M)* → *verify with public key*

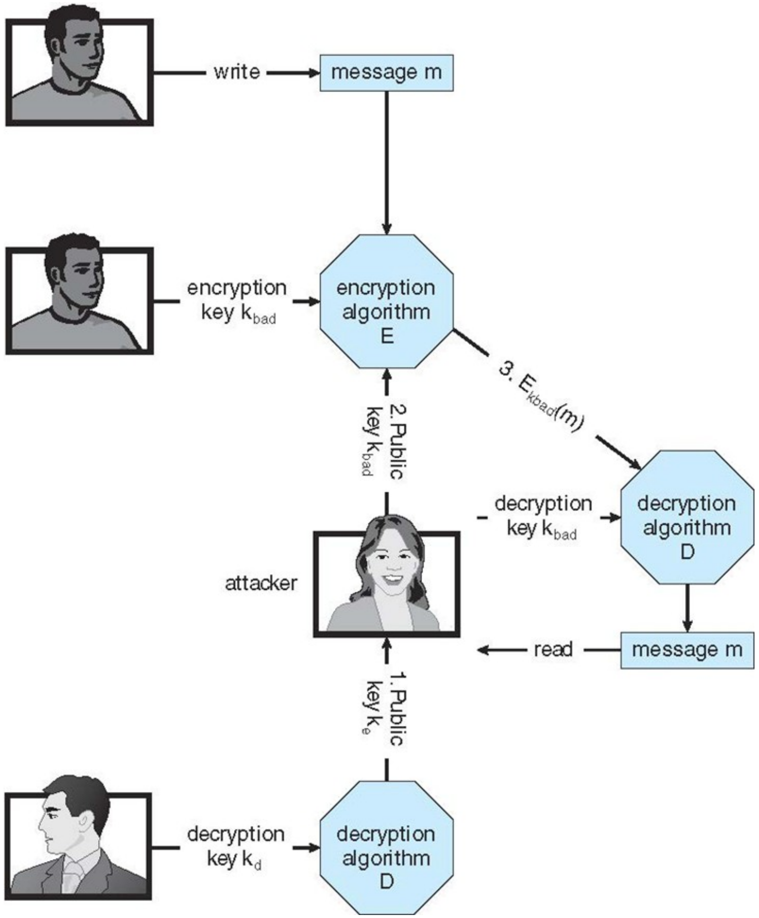
Example: SSH Public Key Authentication



Key Distribution

- Delivery of symmetric key is huge challenge
  - Sometimes done **out-of-band**
- Asymmetric keys can proliferate – stored on **key ring**
  - Even asymmetric key distribution needs care – man-in-the-middle attack

Man-in-the-middle Attack on Asymmetric Cryptography



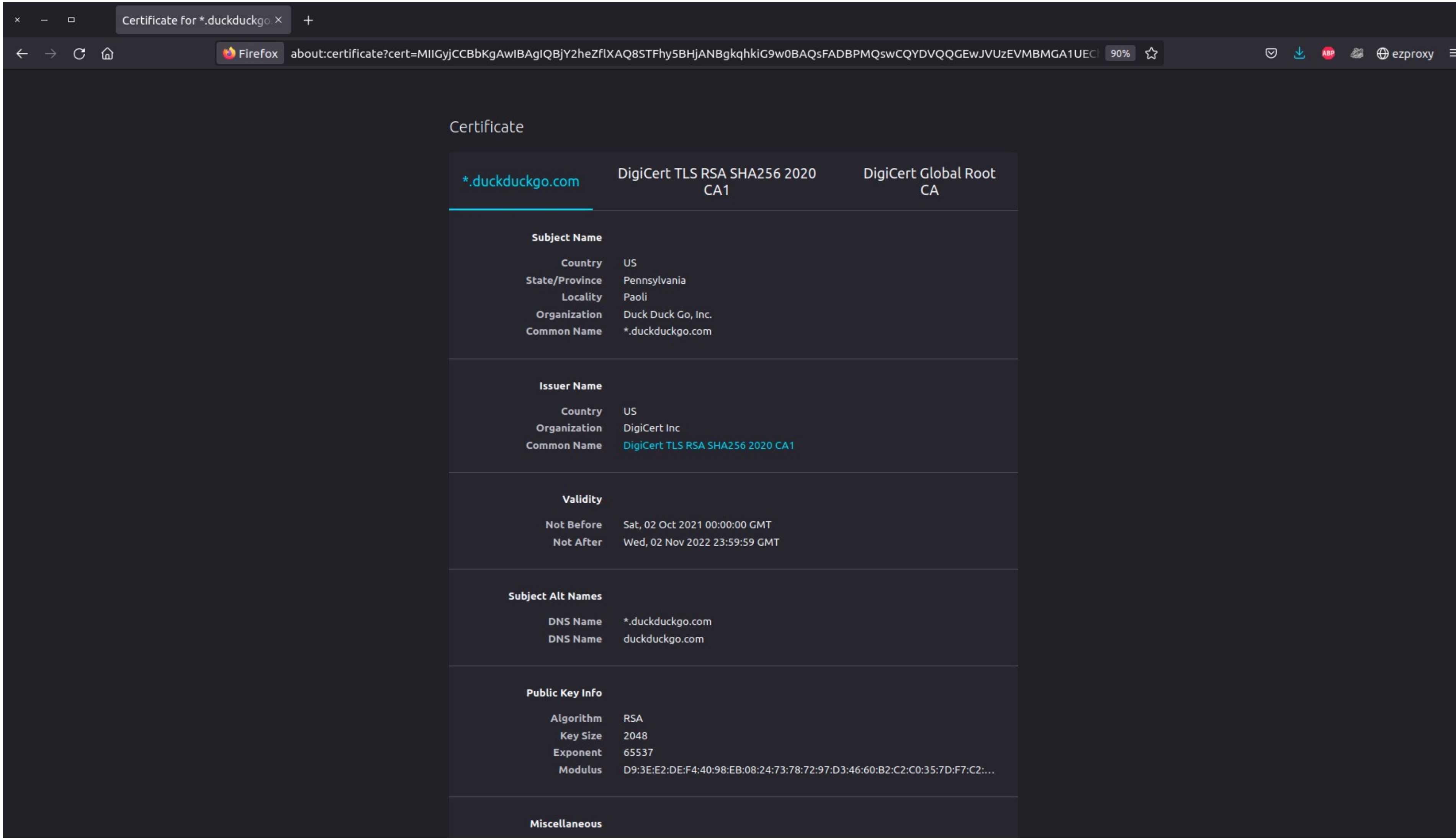
Digital Certificates

- Proof of who or what owns a public key
- **= Public key digitally signed by a trusted party**
- Trusted party receives proof of identification from entity and certifies that public key belongs to entity
- **Certificate authority** are trusted party – their public keys included with web browser distributions
  - They vouch for other authorities via digitally signing their keys, and so on

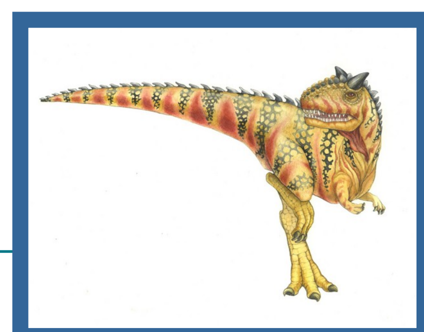
Used for building "chain of trust" in SSL/TLS

Example SSL/TLS certificate

SSL/TLS section in book: Focus on **concepts,** no need to understand/reproduce all details!



16.5 User Authentication

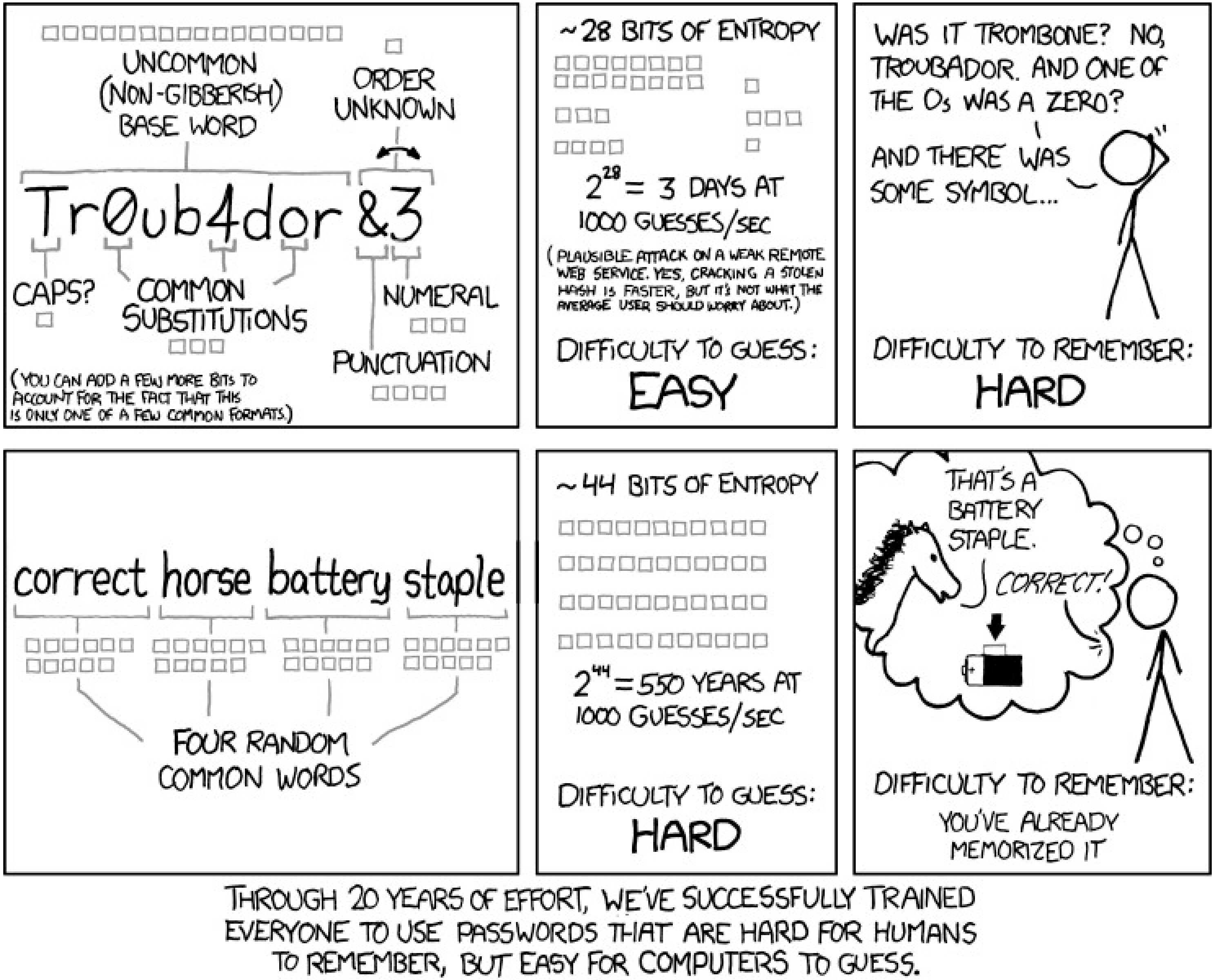


User Authentication [1]

- Crucial to identify user correctly, as protection systems depend on user ID
- User identity most often established through **passwords**
- Passwords must be kept secret
  - Frequent change of passwords
  - History to avoid repeats
  - Use of "non-guessable" passwords
  - Log all invalid access attempts (but not the passwords themselves)
  - Unauthorized transfer

Passwords 1

- **Hash** passwords to avoid having to keep them secret
  - Hash algorithm easy to compute but difficult to invert
  - Only encrypted password stored, never decrypted
  - But keep secret anyway (i.e. Unix uses superuser-only readably file /etc/shadow)
  - Add **“salt”** to avoid the same password being encrypted to the same value
- One-time passwords
  - Use a function based on a seed to compute a password, both user and computer
  - Hardware device / calculator / key fob to generate the password
    - Changes very frequently



https://xkcd.com/936/

Wrap-up Part 1: Security

- **CIA** triad
- Perfect security = impossible → raise the bar on several **levels:** Application, OS, Network
- Beware of malware, viruses
  - → Even benign applications can be hijacked via **code injection** (e.g., buffer overflows)!
- Powerful security primitive: **cryptography**
  - *Symmetric*: e.g., AES + (hash-based) MAC
  - *Asymmetric*: e.g., RSA + digital signatures, certificates